C interfaces to GALAHAD SHA

Jari Fowkes and Nick Gould
STFC Rutherford Appleton Laboratory
Wed Aug 23 2023

1 GALAHAD C package sha **1**

1.1 Introduction . . . 1

1.1.1 Purpose . . . 1

1.1.2 Authors . . . 1

1.1.3 Originally released . . . 1

1.1.4 Method . . . 2

1.1.5 Reference . . . 2

1.1.6 Call order . . . 2

2 File Index **3**

2.1 File List . . . 3

3 File Documentation **5**

3.1 galahad_sha.h File Reference . . . 5

3.1.1 Data Structure Documentation . . . 5

3.1.1.1 struct sha_control_type . . . 5

3.1.1.2 struct sha_inform_type . . . 6

3.1.2 Function Documentation . . . 6

3.1.2.1 sha_initialize() . . . 7

3.1.2.2 sha_reset_control() . . . 7

3.1.2.3 sha_analyse_matrix() . . . 7

3.1.2.4 sha_recover_matrix() . . . 8

3.1.2.5 sha_information() . . . 9

3.1.2.6 sha_terminate() . . . 10

Chapter 1

GALAHAD C package sha

1.1 Introduction

1.1.1 Purpose

Find a **component-wise secant approximation to the Hessian matrix** $H(x)$, for which $(H(x))_{i,j} = \partial f^2(x)/\partial x_i \partial x_j$, $1 \leq i, j \leq n$, using values of the gradient $g(x) = \nabla_x f(x)$ of the function $f(x)$ of n unknowns $x = (x_1, \ldots, x_n)^T$ at a sequence of given distinct $x^{(k)}$, $k \geq 0$. More specifically, given **differences**

$$s^{(k)} = x^{(k+1)} - x^{(k)}$$

and

$$y^{(k)} = g(x^{(k+1)}) - g(x^{(k)})$$

the package aims to find a good estimate B of $H(x)$ for which the secant conditions $Bs^{(k)} = y^{(k)}$ hold approximately for a chosen set of values k. The methods provided take advantage of the entries in the Hessian that are known to be zero.

The package is particularly intended to allow gradient-based optimization methods, that generate iterates $x^{(k+1)} = x^{(k)} + s^{(k)}$ based upon the values $g(x^{(k)})$ for $k \geq 0$, to build a suitable approximation to the Hessian $H(x^{(k+1)})$. This then gives the method an opportunity to accelerate the iteration using the Hessian approximation.

1.1.2 Authors

N. I. M. Gould, STFC-Rutherford Appleton Laboratory, England.

C interface, additionally J. Fowkes, STFC-Rutherford Appleton Laboratory.

Julia interface, additionally A. Montoison and D. Orban, Polytechnique Montréal.

1.1.3 Originally released

April 2013, C interface January 2022.

1.1.4 Method

The package computes the entries in the each row of B one at a time. The entries b_{ij} in row i may be chosen to

$$(1) \quad \underset{b_{i,j}}{\text{minimize}} \quad \sum_{k \in \mathcal{I}_i} \left[\sum_{\text{nonzeros } j} b_{i,j} s_j^{(k)} - y_i^{(k)} \right]^2,$$

where I_i is ideally chosen to be sufficiently large so that (1) has a unique minimizer. Since this requires that there are at least as many $(s^{(k)}, y^{(k)})$ pairs as the maximum number of nonzeros in any row, this may be prohibitive in some cases. We might then be content with a minimum-norm (under-determined) least-squares solution. Or, we may take advantage of the symmetry of the Hessian, and note that if we have already found the values in row j, then the value $b_{i,j} = b_{j,i}$ in (1) is known before we process row i. Thus by ordering the rows and exploiting symmetry we may reduce the numbers of unknowns in future unprocessed rows.

In the analysis phase, we order the rows by constructing the connectivity graph—a graph comprising nodes 1 to n and edges connecting nodes i and j if $h_{i,j}$ is everywhere nonzero—of $H(x)$. The nodes are ordered by increasing degree (that is, the number of edges emanating from the node) using a bucket sort. The row chosen to be ordered next corresponds to a node of minimum degree, the node is removed from the graph, the degrees updated efficiently, and the process repeated until all rows have been ordered. This often leads to a significant reduction in the numbers of unknown values in each row as it is processed in turn, but numerical rounding can lead to inaccurate values in some cases. A useful remedy is to process all rows for which there are sufficient $(s^{(k)}, y^{(k)})$ as before, and then process the remaining rows taking into account the symmetry. That is, the rows and columns are rearranged so that the matrix is in block form

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix},$$

the $(B_{11} \;\; B_{12})$ rows are processed without regard for symmetry but give the $2,1$ block B_{12}^T, and finally the $2,2$ block B_{22} is processed either with the option of exploiting symmetry. More details of the precise algorithms (Algorithms 2.1–2.5) are given in the reference below. The linear least-squares problems (1) themselves are solved by a choice of LAPACK packages.

1.1.5 Reference

The method employed is described in detail in

J. M. Fowkes, N. I. M. Gould and J. A. Scott, Approximating large-scale Hessians using secant equations. Technical Report TR-2023, Rutherford Appleton Laboratory.

1.1.6 Call order

To find the Hessian approximation, functions from the sha package must be called in the following order:

- sha_initialize - provide default control parameters and set up initial data structures
- sha_read_specfile (optional) - override control values by reading replacement values from a file
- sha_analyse_matrix - set up structures needed to construct the Hessian approximation
- sha_recover_matrix - construct the Hessian approximation
- sha_information (optional) - recover information about the solution and solution process
- sha_terminate - deallocate data structures

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

galahad_sha.h 5

Chapter 3

File Documentation

3.1 galahad_sha.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "galahad_precision.h"
#include "galahad_cfunctions.h"
```

Data Structures

- struct sha_control_type
- struct sha_inform_type

Functions

- void sha_initialize (void **data, struct sha_control_type *control, int *status)
- void sha_reset_control (struct sha_control_type *control, void **data, int *status)
- void sha_analyse_matrix (struct sha_control_type *control, void **data, int *status, int n, int ne, const int row[ ], const int col[ ], int *m)
- void sha_recover_matrix (void **data, int *status, int ne, int m, int ls1, int ls2, const real_wp_ s[ ][ls2], int ly1, int ly2, const real_wp_ y[ ][ly2], real_wp_ val[ ], const int precedence[ ])
- void sha_information (void **data, struct sha_inform_type *inform, int *status)
- void sha_terminate (void **data, struct sha_control_type *control, struct sha_inform_type *inform)

3.1.1 Data Structure Documentation

3.1.1.1 struct sha_control_type

control derived type as a C struct

Data Fields

bool	f_indexing	use C or Fortran sparse matrix indexing
int	error	error and warning diagnostics occur on stream error

Data Fields

int	out	general output occurs on stream out
int	print_level	the level of output required. <= 0 gives no output, = 1 gives a one-line summary for every iteration, = 2 gives a summary of the inner iteration for each iteration, >= 3 gives increasingly verbose (debugging) output
int	approximation_algorithm	which approximation algorithm should be used? • 1 : unsymmetric (alg 2.1 in paper) • 2 : symmetric (alg 2.2 in paper) • 3 : composite (alg 2.3 in paper) • 4 : composite 2 (alg 2.4 in paper) • 5 : cautious (alg 2.5 in paper)
int	dense_linear_solver	which dense linear equation solver should be used? • 1 : Gaussian elimination • 2 : QR factorization • 3 : singular-value decomposition • 4 : singular-value decomposition with divide-and-conquer
int	max_sparse_degree	the maximum sparse degree if the combined version is used
int	extra_differences	if available use an addition extra_differences differences
bool	space_critical	if space is critical, ensure allocated arrays are no bigger than needed
bool	deallocate_error_fatal	exit if any deallocation fails
char	prefix[31]	all output lines will be prefixed by .prefix(2:LEN(TRIM(.prefix))-1) where .prefix contains the required string enclosed in quotes, e.g. "string" or 'string'

3.1.1.2 struct sha_inform_type

inform derived type as a C struct

Data Fields

int	status	return status. See SHA_solve for details
int	alloc_status	the status of the last attempted allocation/deallocation.
int	max_degree	the maximum degree in the adgacency graph.
int	approximation_algorithm_used	which approximation algorithm has been used
int	differences_needed	the number of differences that will be needed.
int	max_reduced_degree	the maximum reduced degree in the adgacency graph.
int	bad_row	a failure occured when forming the bad_row-th row (0 = no failure).
char	bad_alloc[81]	the name of the array for which an allocation/deallocation error occurred.

3.1.2 Function Documentation

3.1.2.1 sha_initialize()

```
void sha_initialize (
             void ** data,
             struct sha_control_type * control,
             int * status )
```

Set default control values and initialize private data

Parameters

in,out	*data*	holds private internal data
out	*control*	is a struct containing control information (see sha_control_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The initialization was succesful.

3.1.2.2 sha_reset_control()

```
void sha_reset_control (
             struct sha_control_type * control,
             void ** data,
             int * status )
```

Reset control parameters after import if required.

Parameters

in	*control*	is a struct whose members provide control paramters for the remaining prcedures (see sha_control_type)
in,out	*data*	holds private internal data
in,out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are: • 0. The import was succesful.

3.1.2.3 sha_analyse_matrix()

```
void sha_analyse_matrix (
             struct sha_control_type * control,
             void ** data,
             int * status,
             int n,
             int ne,
             const int row[],
```

```
            const int col[],
            int * m )
```

Import structural matrix data into internal storage prior to solution

Parameters

in	*control*	is a struct whose members provide control paramters for the remaining prcedures (see sha_control_type)
in,out	*data*	holds private internal data
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are: • 0. The import and analysis were conducted succesfully. • -1. An allocation error occurred. A message indicating the offending array is written on unit control.error, and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -2. A deallocation error occurred. A message indicating the offending array is written on unit control.error and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -3. The restrictions n $>$ 0 or ne $>=$ 0 has been violated.
in	*n*	is a scalar variable of type int, that holds the number of rows in the symmetric matrix H.
in	*ne*	is a scalar variable of type int, that holds the number of entries in the upper triangular part of H in the sparse co-ordinate storage scheme
in	*row*	is a one-dimensional array of size ne and type int, that holds the row indices of the upper triangular part of H in the sparse co-ordinate storage scheme
in	*col*	is a one-dimensional array of size ne and type int, that holds the column indices of the upper triangular part of H in sparse row-wise storage scheme.
out	*m*	is a scalar variable of type int, that holds the minimum number of $(s^(k), y^(k))$ pairs that will be needed to recover a good Hessian approximation

3.1.2.4 sha_recover_matrix()

```
void sha_recover_matrix (
            void ** data,
            int * status,
            int ne,
            int m,
            int ls1,
            int ls2,
            const real_wp_ s[][ls2],
            int ly1,
            int ly2,
            const real_wp_ y[][ly2],
            real_wp_ val[],
            const int precedence[] )
```

Form and factorize the symmetric matrix A.

Parameters

in,out	*data*	holds private internal data
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are: • 0. The factors were generated succesfully. • -1. An allocation error occurred. A message indicating the offending array is written on unit control.error, and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -2. A deallocation error occurred. A message indicating the offending array is written on unit control.error and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -3. The restrictions n $>$ 0 or ne $>=$ 0 has been violated.
in	*ne*	is a scalar variable of type int, that holds the number of entries in the upper triangular part of the symmetric matrix H.
in	*m*	is a scalar variable of type int, that holds the number of (s, y) pairs that are available.
in	*s*	is a two-dimensional array of size (ls1,ls2) and type double, that holds the values of the vectors $s^{(k)}$. Component i, k holds $s_i^{(k)}$.
in	*ls1*	is a scalar variable of type int, that holds the leading dimension of the array s.
in	*ls2*	is a scalar variable of type int, that holds the trailing dimension of the array s.
in	*y*	is a two-dimensional array of size (ly1,ly2) and type double, that holds the values of the vectors $y^{(k)}$. Component i, k holds $y_i^{(k)}$.
in	*lu1*	is a scalar variable of type int, that holds the leading dimension of the array y.
in	*ly2*	is a scalar variable of type int, that holds the trailing dimension of the array y.
out	*val*	is a one-dimensional array of size ne and type double, that holds the values of the entries of the upper triangular part of the symmetric matrix H in the sparse coordinate scheme.
in	*precedence*	is a one-dimensional array of size m and type int, that holds the preferred order of access for the pairs $(s^(k), y^(k))$. The k-th component of order specifies the column number of s and y that will be used as the k-th most favoured. precedence need not be set if the natural order, $k, k = 1, ...,$ m, is desired, and this case precedence should be NULL.

3.1.2.5 sha_information()

```
void sha_information (
            void ** data,
            struct sha_inform_type * inform,
            int * status )
```

Provides output information

Parameters

in,out	*data*	holds private internal data

Parameters

out	*inform*	is a struct containing output information (see sha_inform_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The values were recorded succesfully

3.1.2.6 sha_terminate()

```
void sha_terminate (
             void ** data,
             struct sha_control_type * control,
             struct sha_inform_type * inform )
```

Deallocate all internal private storage

Parameters

in,out	*data*	holds private internal data
out	*control*	is a struct containing control information (see sha_control_type)
out	*inform*	is a struct containing output information (see sha_inform_type)